IBM RPG

From Wikipedia, the free encyclopedia

RPG is a high-level programming language (HLL) for business applications. Originally the initials stood for **Report Program Generator**, but currently it is referred to only as "RPG". IBM is the creator and primary vendor of RPG, but the language is available from other mainframe and microcomputer manufacturers, including Unisys. The latest version of RPG is **RPG IV** (aka ILE RPG) on IBM's Power i servers; it inherits the System i Integrated Language Environment's features such as prototyped functions and procedures, static and dynamic binding, access to C routine libraries, dynamic link libraries, and fully recursive and re-entrant modular code.

RPG
Report Program Generator

Paradigm	Multi-paradigm
Appeared in	1959
Developer	IBM
Stable release	RPG IV version 6 release 1 (January 29, 2008)
Typing discipline	Strong, static
Dialects	RPG, RPG II, RPG III, RPG 400, RPG IV, RPG/ILE; RPG/Free, Baby/36, Baby/400, Lattice RPG
Influenced by	9PAC, FARGO
Influenced	RPG II
OS	CPF, SSP, OS/400, OS/VS1, z/OS, VS/9, OpenVMS, Burroughs MCP, Windows

Contents

Overview

RPG (aka RPG IV aka RPGLE) is a popular programming language of the IBM Power i platform. Originally created as a report writer tool on the card-input IBM mainframes in the 1960s, RPG has since evolved into an HLL equivalent to IBM's COBOL and PL/I.

An RPG program typically starts off with File Specifications, listing all files being written to, read from or updated, followed by Data Definition Specifications containing program elements such as Data Structures and dimensional arrays, much like a "Working-Storage" section of a COBOL program or var statements in a Pascal program. This is followed by Calculation Specifications, which contain the actual meat of the code. Output Specifications can follow which can be used to determine the layout of other files or reports. Alternatively files, some data structures and reports can be defined externally, mostly eliminating the need to hand code input and output specifications.

In the early days of RPG, its major strength was known as the **program cycle**: every RPG program executes within an implied loop, which can apply the program to every record of a file. At that time each record (individual punched card) would be compared to each line in the program, which would act upon the record, or

not, based upon whether that line had an "indicator" turned "on" or "off" — from a set of logical variables numbered 01–99 for user-defined purposes, or other smaller sets based upon record, field, or report processing functions.

Alternatively, the cycle can make an interactive program continue to run until explicitly stopped.

Today, most RPG programmers avoid using the cycle in favor of controlling the flow of the program with standard looping constructs. The concept of level breaks and matching records is unique to the RPG II language. It was originally developed with card readers in mind. RPG III adds some interesting constructs, but the original RPG language is difficult to beat assuming the developer embraces all of the available constructs and features.

History

RPG is one of the few languages created for punched card machines that is still in common use today. This is because the language has evolved considerably over time. It was originally developed by IBM in the 1960s. The name *Report Program Generator* was descriptive of the purpose of the language: generation of reports from data files, including matching record and sub-total reports.

FARGO (Fourteen-o-one [IBM 1401] Automatic Report Generation Operation) was the predecessor to RPG. Both FARGO and RPG were intended to facilitate ease of transition for IBM tabulating machine unit record equipment technicians to the new IBM 1401 series of computers.

Tab machine technicians were accustomed to plugging wires into control panels to implement input, output, control and counter operations (add, subtract, multiply, divide). Tab machines programs were executed by impulses emitted in a machine cycle; hence, FARGO and RPG emulated the notion of the machine cycle with the program cycle. RPG was superior to and rapidly replaced FARGO as the report generator program of choice.

The alternative languages generally available at the time were Assembler, COBOL or FORTRAN. COBOL was a natural language-like business oriented language and FORTRAN was a language that facilitated mathematical applications. Other languages of the era included ALGOL and Autocoder and a few years later PL/1. Assembler and COBOL were more common in mainframe business operations (System/360 models 30 and above) and RPG was more commonly used by customers who were in transition from tabulating equipment (System/360 model 20).

Language evolution

RPG II was introduced with the System/3 series of computers. It was later used on System/32, System/34, and System/36, with an improved version of the language. ICL also produced a version on its VME/K operating system

RPG III was created for the System/38 and its successor the AS/400 . RPG III significantly departed from the original language, providing modern structured constructs like IF-ENDIF blocks, DO loops, and subroutines (RPG2 Supported Subroutines). RPG III was also available for larger systems including the IBM System/370 mainframe running OS/VS1. It was also available from Unisys for the VS/9 operating system running on the Univac 90/60 mainframe.

DE/RPG or Data Entry RPG was exclusively available on the IBM 5280 series of data-entry workstations in the early 80s. It was similar to RPG III but lacking external Data Descriptions (DDS) to describe data(files) like on the System/38 and its successors. Instead, the DDS part had to be included into the RPG source itself.

RPG/400 with a much cleaner syntax, and tighter integration with the integrated database. This language became the mainstay of development on the AS/400, and its editor was a simple line editor with prompt templates for each specification (type of instruction).

RPG IV (aka RPGLE, aka RPG/ILE) was released in 1994 and the name, officially, was no longer an initialism. RPG IV offered a greater variety of expressions within its new Extended Factor-2 Calculation Specification.

. . .

RPG IV, a modern language

In 2001, with the release of OS/400 V5R1, RPG IV offered even greater freedom for calculations than offered by the Extended Factor-2 Calculation Specification: a *free-format* text-capable source entry, as an alternative to the original column-dependent source format. The "/FREE" calculation does not require the operation code to be placed in a particular column; the operation code is optional for the EVAL and CALLP operations; and syntax generally more closely resembles that of mainstream, general-purpose programming languages.

Today, RPG IV is a considerably more robust language. Editing can still be done via the simple editor or it can be edited via PC using IBM's Websphere Development Studio now named RDi (Rational Development Studio for i) a customized implementation of Eclipse. IBM is continually extending its capabilities and adding more built-in functions (BIFs). It has the ability to link to Java objects,[1] and i5/OS APIs; it can be used to write CGI programs with the help of IBM's Cgidev2 web toolkit,[2] RPGLIB (a collection of hundreds of pre-written RPG IV routines) (http://www.rpglib.com) , CGILIB (http://www.rpglib.com) , the RPG Toolbox, and other commercial Web enabled packages. Even with the changes, it retains a great deal of backward compatibility, so an RPG program written 37 years ago could run today with little or no modification.

Furthermore, with the implementation of the SQL Precompiler, current RPG developers can take advantage of IBM's cost-based SQE (SQL Query Engine). As opposed to the traditional F-Spec approach, where a developer identified a specific access path to a data set, a developer can now implement standard embedded SQL statements directly in the program. When compiled, the SQL Precompiler transforms the invalid embedded SQL statements into valid RPG statements that call the database manager programs that ultimately implement the query request.

OS/400 was later renamed i5/OS to correspond with the new IBM System i5 branding initiative; the 5 was later dropped in favor of just System i. In March 2008 i5/OS was renamed IBM i as part of the Power Systems consolidation of System i and System p product lines. The new Power Systems also adopt more mainstream version numbers, substituting 6.1 for the twenty year old V1R1M0 notation. The latest release is now referred to as IBM i 7.1 and fully supports the RPG IV language, as well as many others. WebSphere Development Studio Client (WDSC) is now referred to as Rational Developer for i (RDi), of which three product levels are available. They are called Rational Developer for i (RDi), RDi Service Oriented Architecture (RDi SOA), and Rational Application Developer (RAD). The new lineup provides in more granular packaging all of the development tools and support previously offered by WDSC and WDSC Advanced Edition.

Example code

The following program receives a customer number as an input parameter and returns the name and address as output parameters.

```
     * Historically RPG is columnar in nature, though free-formatting
     * is allowed under particular circumstances.
     * The purpose of various lines code are determined by a
     * letter code in column 6.
```

```
     * An asterisk (*) in column 7 denotes a comment line

     * "F" (file) specs define files and other i/o devices
    F ARMstF1   IF   E       K     Disk    Rename(ARMST:RARMST)

     * "D" specs are used to define variables
    D pCusNo          S              6p 0
    D pName           S             30a
    D pAddr1          S             30a
    D pAddr2          S             30a
    D pCity           S             25a
    D pState          S              2a
    D pZip            S             10a

     * "C" (calculation) specs are used for executable statements
     * Parameters are defined using plist and parm opcodes
    C     *entry        plist
    C                   parm                    pCusNo
    C                   parm                    pName
    C                   parm                    pAddr1
    C                   parm                    pAddr2
    C                   parm                    pCity
    C                   parm                    pState
    C                   parm                    pZip

     * The "chain" command is used for random access of a keyed file
    C     pCusNo        chain     ARMstF1

     * If a record is found, move fields from the file into parameters
    C                   if        %found
    C                   eval      pName  = ARNm01
    C                   eval      pAddr1 = ARAd01
    C                   eval      pAddr2 = ARAd02
    C                   eval      pCity  = ARCy01
    C                   eval      pState = ARSt01
    C                   eval      pZip   = ARZp15
    C                   endif

     * RPG makes use of switches.  One switch "LR" originally stood for "last record"
     * LR actually flags the program and its dataspace as removable from memory.
    C                   eval      *InLR = *On
```

The same program using free calculations:

```
     * "F" (file) specs define files and other i/o devices
    FARMstF1   IF   E        K     Disk    Rename(ARMST:RARMST)

     * "D" specs are used to define variables and parameters
     * The "prototype" for the program is in a separate file
     * allowing other programs to call it
     /copy cust_pr
     * The "procedure interface" describes the *ENTRY parameters
    D getCustInf      PI
    D  pCusNo                        6p 0   const
    D  pName                        30a
    D  pAddr1                       30a
    D  pAddr2                       30a
    D  pCity                        25a
    D  pState                        2a
    D  pZip                         10a
     /free
       // The "chain" command is used for random access of a keyed file
       chain pCusNo ARMstF1;

       // If a record is found, move fields from the file into parameters
       if %found;
          pName  = ARNm01;
```

```
          pAddr1 = ARAd01;
          pAddr2 = ARAd02;
          pCity  = ARCy01;
          pState = ARSt01;
          pZip   = ARZp15;
        endif;

      // RPG makes use of switches.  One switch "LR" originally stood for "last record"
      // LR actually flags the program and its dataspace as removable from memory.
        *InLR = *On;
      /end-free
```

The same program using free calculations and embedded SQL:

```
      * "D" specs are used to define variables and parameters
      * The "prototype" for the program is in a separate file
      * allowing other programs to call it
      /copy cust_pr
      * The "procedure interface" describes the *ENTRY parameters
     D getCustInf      PI
     D  pCusNo                        6p 0   const
     D  pName                        30a
     D  pAddr1                       30a
     D  pAddr2                       30a
     D  pCity                        25a
     D  pState                        2a
     D  pZip                         10a
      /free
        exec sql select ARNm01, ARAd01, ARAd02, ARCy01, ARSt01, ARZp15
                  into  :pName, :pAddr1, :pAddr2, :pCity, :pState, :pZip
                  from   ARMstF1
                  where  ARNo01 = :pCusNo
                  for fetch only
                  fetch first 1 row only
                  optimize for 1 row
                  with CS;

      // RPG makes use of switches.  One switch "LR" originally stood for "last record"
      // LR actually flags the program and its dataspace as removable from memory.
        *InLR = *On;
      /end-free
```

Platforms

As stated above, the RPG programming language originally was introduced by IBM for their proprietary 1401, /360, /3, /32, /34, /36, /38 AS/400 and System i systems. There have also been implementations for the Digital VAX, Sperry Univac BC/7, Univac system 80, Siemens BS2000, Burroughs B1700, Hewlett Packard HP3000, ICL 2900 series, Honeywell 6220 and 2020, Four-Phase IV/70 and IV/90 series and WANG VS, as well as miscellaneous compilers & runtime environments for Unix-based systems (INFINITE 36 (formerly UNIBOL36) (http://www.infinitesoftware.com/solutions/Infinite36.asp)) and PCs (Baby/400, Lattice-RPG).

The latest platform to receive an RPG compiler is Windows .Net through the WINRPG compiler. This version contains extensions to RPG IV beyond that of the base IBM compiler. These extensions provide Windows and .Net hooks.

RPG II applications are still supported under the IBM zVSE operating system, HP MPE operating system on HP3000 and the OpenVMS operating system on VAX, Alpha, and Unisys MCP.

Further reading

- McGee, W.C. (September 1981). "Data Base Technology" (http://www.research.ibm.com/journal/rd/255/ibmrd2505O.pdf) (PDF). *IBM Journal of Research and Development* (IBM) **25** (5, 25th Anniversary Issue): 514. http://www.research.ibm.com/journal/rd/255/ibmrd2505O.pdf. "...original report program generator was the IBM Report Program Generator introduced in the early 1960s for the IBM 1401 computer. It was patterned after the SHARE 9PAC system...".
- "9PAC, Report Generator" (http://hopl.murdoch.edu.au/showlanguage2.prx?exp=35) . History of Programming Languages (HOPL), Murdoch University, AU. 2006. http://hopl.murdoch.edu.au/showlanguage2.prx?exp=35.
- "RPG, Report Program Generator" (http://hopl.murdoch.edu.au/showlanguage2.prx?exp=207) . History of Programming Languages (HOPL), Murdoch University, AU. 2006. http://hopl.murdoch.edu.au/showlanguage2.prx?exp=207.
- DuCharme, Bob (2006-02-26). "Pulling data out of computers in the mid-twentieth and early twenty-first centuries" (http://www.snee.com/bobdc.blog/2006/02/pulling_data_out_of_computers.html) . http://www.snee.com/bobdc.blog/2006/02/pulling_data_out_of_computers.html.
- Shelly, Gary B.; Thomas J. Cashman (1977). *Introduction to Computer Programming RPG*. Fullerton, California: Anaheim Publishing Company. ISBN 0-88236-225-9.

References

1. ^ http://publib.boulder.ibm.com/iseries/v5r1/ic2924/books/c0918160.pdf
2. ^ http://www.easy400.net/cgidev2/start

External links

- RPG IV for beginners (http://tutorialindia.com/ile_rpg_iv/index.php) - A basic tutorial for beginners is available here.
- Midrange.com (http://www.midrange.com/) - A large amount of code examples are available here
- RPG (IV) World (http://www.rpgworld.com) - An RPG discussion forum and a huge amount of code examples.
- RPG OPEN (http://www.rpgopen.com) - A free open source service program with add-on subprocedures written in RPG IV for RPG IV.
- IBM (1964) (PDF). *IBM 1401 RPG manual* (http://www.bitsavers.org/pdf/ibm/140x/C24-3261-1_1401_diskRPG.pdf) . C24-3261-1. http://www.bitsavers.org/pdf/ibm/140x/C24-3261-1_1401_diskRPG.pdf.
- IBM (2008) (PDF). *ILE RPG Programmer's Guide* (http://publib.boulder.ibm.com/infocenter/iseries/v6r1m0/topic/rzasc/sc092507.pdf) . SC09-2507-07. http://publib.boulder.ibm.com/infocenter/iseries/v6r1m0/topic/rzasc/sc092507.pdf.
- RPG for COBOL programmers (http://www.jcmigrations.com/training.htm#rpg) - Article
- RPG II for MVS, OS/390 and z/OS (http://gsf-soft.com/Documents/RPG-ZOS.shtml) - Status of the IBM RPG II product in z/OS

Retrieved from "http://en.wikipedia.org/wiki/IBM_RPG"
Categories: Data-centric programming languages | Procedural programming languages | IBM software | 1959 software | Programming languages created in the 1950s